A Theory of Scientific Programming Efficacy

Elizaveta Pertseva, Melinda Chang, Ulia Zaman, **Michael Coblenz**

UC San Diego
JACOBS SCHOOL OF ENGINEERING
Department of Computer Science and Engineering

Our Perspective

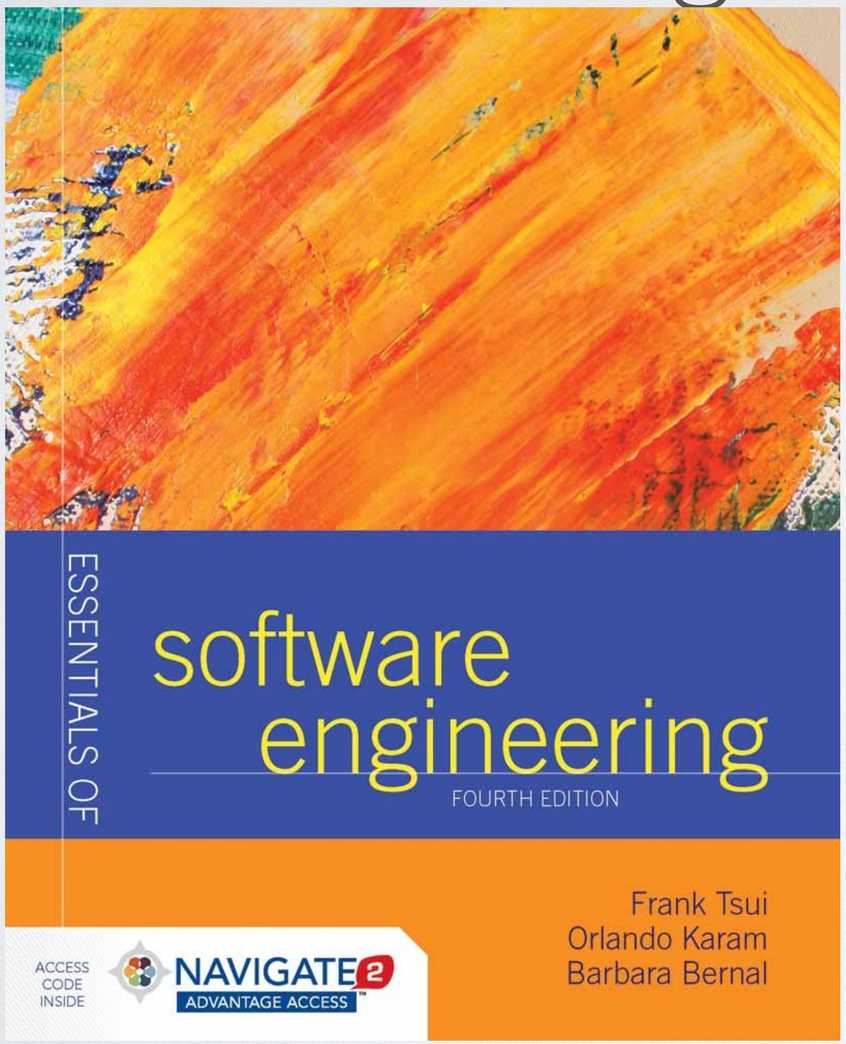
- Programming languages
- Software engineering
- Human-computer interaction
- Overall research question: how can we help people who write software be more effective?

Programming Languages Researchers Care About…

- Proving that programs are correct (what does "correct" mean?)
- Proving that programs don't have specific classes of bugs
- Making it easy to reason about programs
- Making programs run efficiently (faster, less energy)

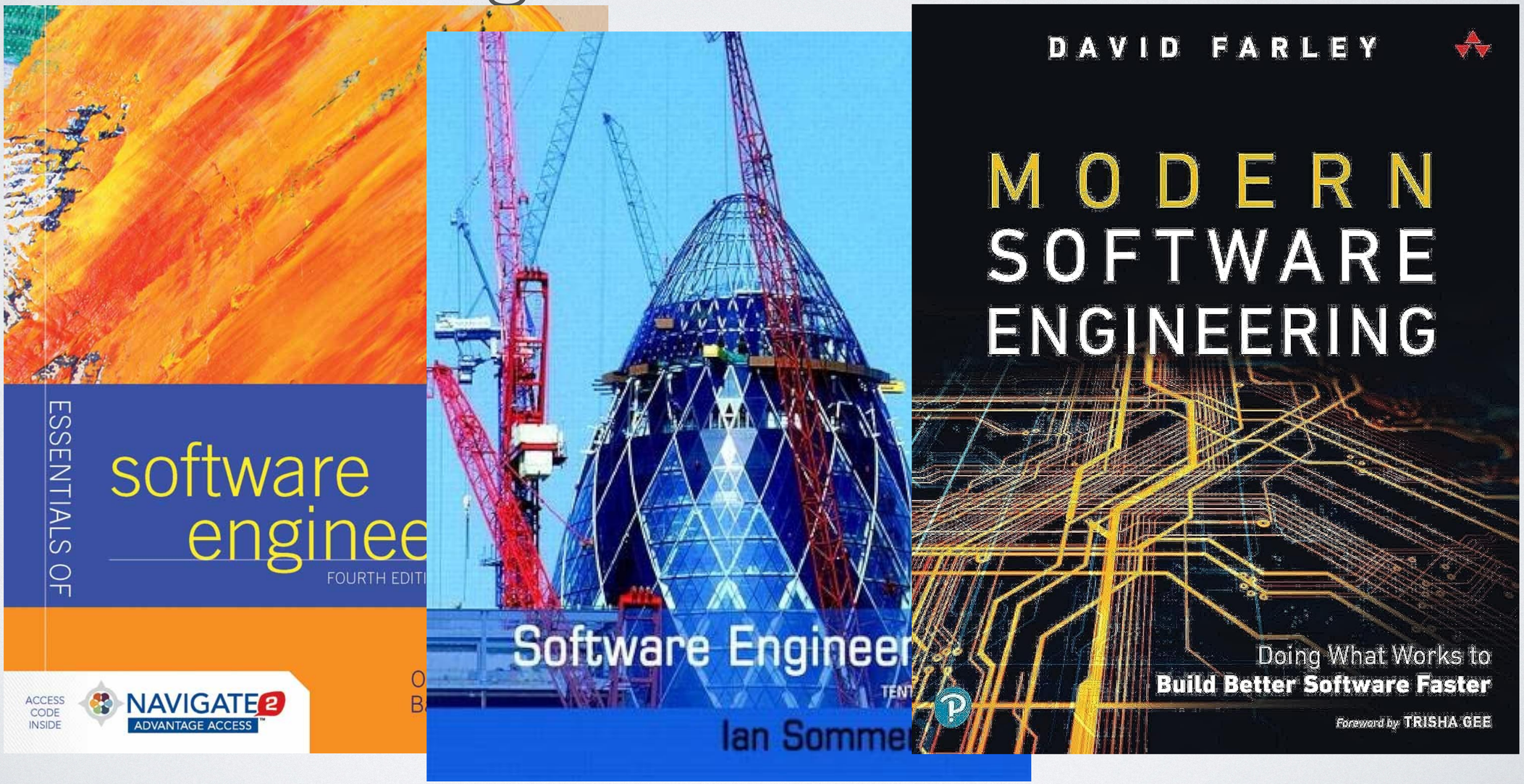
What Should We Tell Scientists About How To Engineer Software?

What Should We Tell Scientists About How To Engineer Software?



What Should We Tell Scientists About How To Engineer Software?

What Should We Tell Scientists About How To Engineer Software?



Assumptions of SE Research

- Software engineers:
  - have extensive training in programming
  - care about software
  - work in large teams to build large artifacts over a long time

For Scientists…

- Software is secondary (to results, papers)
- Programming background is inconsistent and potentially minimal
- Many small (< 10 KLOC) projects
  - that may leverage earlier projects

What Practices Lead to Effective Scientific Software Engineering?

- Interviewed 25 scientists about practices, challenges
- Used techniques from *grounded theory* to analyze transcripts
  - Qualitative research: goal is to hypothesize a theory, identify opportunities

Participants

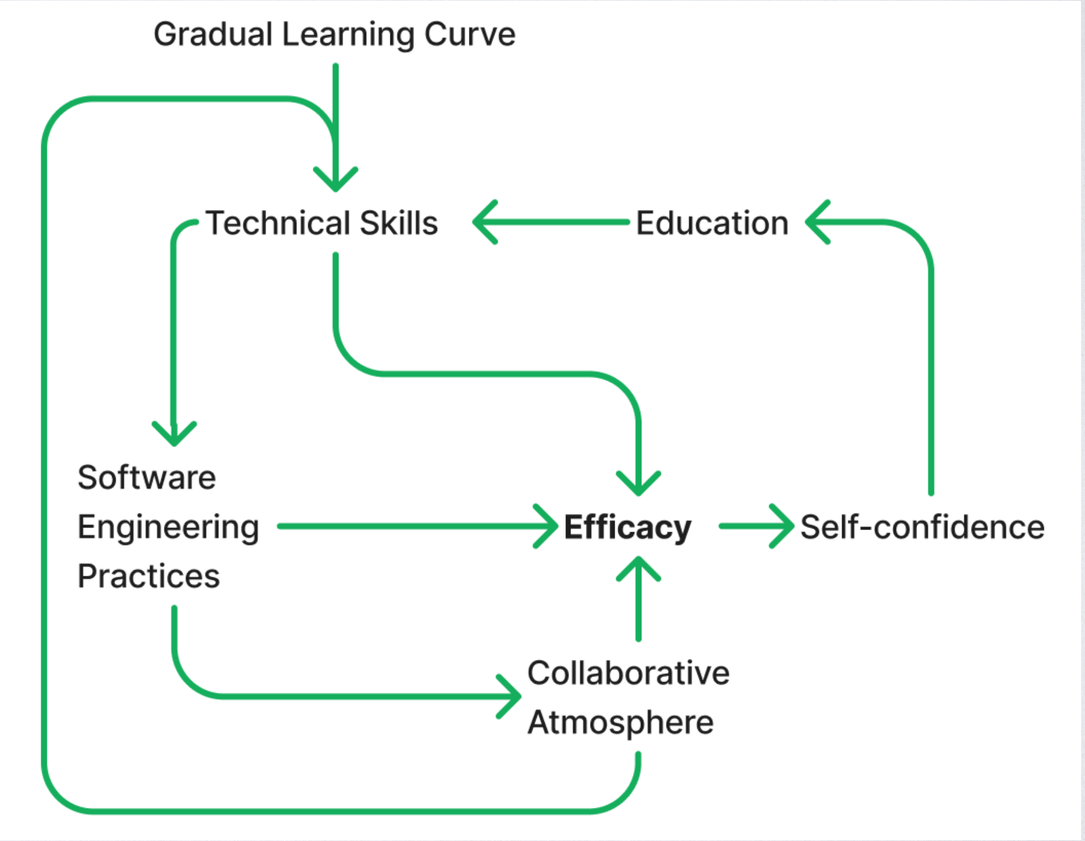
- 20 scientists
- 5 support staff

#	Field	Position	Education	Yrs.
P1	Oceanogr.	Post Doc	PhD Oceanogr.	8
P2	Oceanogr.	Proj. Scientist	PhD Oceanogr.	20
P3	Oceanogr.	Professor	PhD Oceanogr.	-
P4	Oceanogr.	PhD Student	BS Atmos. Sci.	5
P5	Econ.	Researcher	PhD Economics	22
P6	Oceanogr.	Researcher	PhD Oceanogr.	-
P7	Math.	Researcher	PhD Math.	10
P8*	Geosciences	Engineer	MS DSE	25
P9	Physics	Professor	PhD Physics	30
P10	Linguistics	PhD Student	MS Linguistics	-
P11	Climate Sci.	Post Doc	PhD Atmos. Sci.	-
P12*	HPC	User Support	PhD Aerosp. Eng.	-
P13*	HPC, Clim. Sci.	Research Staff	PhD Math.	45
P14	Bioinform.	Professor	PhD Comp. Sci.	20
P15	Clim. Sci.	PhD Student	BS Physics	5
P16	Oceanogr.	Lab. Director	PhD Physics	-
P17	Oceanogr.	PhD Student	BA Physics	4
P18	Oceanogr.	MS Student	BS Eng. Physics	6
P19	Econ.	Pre Doc	BS Econom.	3
P20	Econ.	PhD Candidate	BS Economics	-
P21	Bioinform.	PhD Student	BS Biology	3
P22	Glob. Policy.	Post Doc	PhD Atmos. Sci.	-
P23*	Research IT	SI Engineer	PhD Physics	20
P24	Oceanogr.	Post Doc	PhD Earth Sci.	11
P25*	Data Curation	Librarian	MLIS	7

Results

Efficacy

- Identified six factors that relate to efficacy
- Positive feedback loop: Self-confidence inspires more education, which promotes skills



SE Practices

SE Practices

- Testing often ad hoc and visualization-based (e.g., 9 tried unit tests but most found it futile)

SE Practices

- Testing often ad hoc and visualization-based (e.g., 9 tried unit tests but most found it futile)
- Version control used primarily for collaboration (sometimes with ad hoc methods)

SE Practices

- Testing often ad hoc and visualization-based (e.g., 9 tried unit tests but most found it futile)
- Version control used primarily for collaboration (sometimes with ad hoc methods)
  - "Before starting to use version control. . . six months pass, and you realize that some analysis that you have done before. . . [is] actually sort of useful, right? And then if you have to. . . rewrite it every time, it can get quite annoying."

SE Practices

- Testing often ad hoc and visualization-based (e.g., 9 tried unit tests but most found it futile)
- Version control used primarily for collaboration (sometimes with ad hoc methods)
  - “Before starting to use version control. . . six months pass, and you realize that some analysis that you have done before. . . [is] actually sort of useful, right? And then if you have to. . . rewrite it every time, it can get quite annoying."
- No structured processes (e.g. agile)

SE Practices

- Testing often ad hoc and visualization-based (e.g., 9 tried unit tests but most found it futile)
- Version control used primarily for collaboration (sometimes with ad hoc methods)
  - "Before starting to use version control. . . six months pass, and you realize that some analysis that you have done before. . . [is] actually sort of useful, right? And then if you have to. . . rewrite it every time, it can get quite annoying."
- No structured processes (e.g. agile)
- 2/20 scientists in ongoing collaboration with multiple programmers

SE Practices

- Testing often ad hoc and visualization-based (e.g., 9 tried unit tests but most found it futile)
- Version control used primarily for collaboration (sometimes with ad hoc methods)
  - "Before starting to use version control. . . six months pass, and you realize that some analysis that you have done before. . . [is] actually sort of useful, right? And then if you have to. . . rewrite it every time, it can get quite annoying."
- No structured processes (e.g. agile)
- 2/20 scientists in ongoing collaboration with multiple programmers
- Major opportunities here!

Collaborative Atmosphere

Collaborative Atmosphere

- 6/20 worked on solo + advisor projects

Collaborative Atmosphere

- 6/20 worked on solo + advisor projects
  - "I had asked [my advisor] 'Can I do Python?' and he was like, 'You can do whatever, but I can only help you in MATLAB.' I'm gonna need help. So yeah, I decided to do it in MATLAB."

Collaborative Atmosphere

- 6/20 worked on solo + advisor projects
  - "I had asked [my advisor] 'Can I do Python?' and he was like, 'You can do whatever, but I can only help you in MATLAB.' I'm gonna need help. So yeah, I decided to do it in MATLAB."
- Reliance on tech support (e.g. at supercomputing centers)

Collaborative Atmosphere

- 6/20 worked on solo + advisor projects
  - "I had asked [my advisor] ‘Can I do Python?’ and he was like, ‘You can do whatever, but I can only help you in MATLAB.’ I’m gonna need help. So yeah, I decided to do it in MATLAB."
- Reliance on tech support (e.g. at supercomputing centers)
  - But tech support tempers advice according to scientists' skills

Collaborative Atmosphere

- 6/20 worked on solo + advisor projects
  - "I had asked [my advisor] ‘Can I do Python?’ and he was like, ‘You can do whatever, but I can only help you in MATLAB.’ I’m gonna need help. So yeah, I decided to do it in MATLAB."
- Reliance on tech support (e.g. at supercomputing centers)
  - But tech support tempers advice according to scientists' skills
- Sense of belonging to community is important for engagement, but scientists don't identify as software engineers

Self-Confidence

Self-Confidence

- Motivation theory: if the goal is performance (not intellectual fulfillment), self-confidence motivates learning

Self-Confidence

- Motivation theory: if the goal is performance (not intellectual fulfillment), self-confidence motivates learning
- Failures may de-motivate learning

Self-Confidence

- Motivation theory: if the goal is performance (not intellectual fulfillment), self-confidence motivates learning
- Failures may de-motivate learning
- P21: bash is "little bit stressful sometimes . . . I accidentally used a recursive chmod on my personal computer, and I locked myself out of it . . . it has so much power over your actual system."

Self-Confidence

- Motivation theory: if the goal is performance (not intellectual fulfillment), self-confidence motivates learning
- Failures may de-motivate learning
- P21: bash is "little bit stressful sometimes . . . I accidentally used a recursive chmod on my personal computer, and I locked myself out of it . . . it has so much power over your actual system."
  - Currently waits for job to finish rather than learning parallelization. "At first, I thought it was going to be really easy . . . I haven't used it yet, but I know it exists and will be helpful."

Self-Confidence

- Motivation theory: if the goal is performance (not intellectual fulfillment), self-confidence motivates learning
- Failures may de-motivate learning
- P21: bash is "little bit stressful sometimes . . . I accidentally used a recursive chmod on my personal computer, and I locked myself out of it . . . it has so much power over your actual system."
  - Currently waits for job to finish rather than learning parallelization. "At first, I thought it was going to be really easy . . . I haven't used it yet, but I know it exists and will be helpful."
- 17/20 reported apprehension or anxiety about programming

Self-Confidence

- Motivation theory: if the goal is performance (not intellectual fulfillment), self-confidence motivates learning
- Failures may de-motivate learning
- P21: bash is "little bit stressful sometimes . . . I accidentally used a recursive chmod on my personal computer, and I locked myself out of it . . . it has so much power over your actual system."
  - Currently waits for job to finish rather than learning parallelization. "At first, I thought it was going to be really easy . . . I haven't used it yet, but I know it exists and will be helpful."
- 17/20 reported apprehension or anxiety about programming
- 18/20 reported guilt over not following SE guidelines. "Oh I probably should be doing that . . . but I haven't had a reason to do [it]."

Education

Education

- 15/20: lack of education is a primary problem

Education

- 15/20: lack of education is a primary problem
  - 10 of these had taken programming classes, but they weren't enough

Education

- 15/20: lack of education is a primary problem
  - 10 of these had taken programming classes, but they weren't enough
- Current approaches: workshops (7/20), e.g. Software Carpentry

Education

- 15/20: lack of education is a primary problem
  - 10 of these had taken programming classes, but they weren't enough
- Current approaches: workshops (7/20), e.g. Software Carpentry
- But workshops are too shallow: "P10 explained, "[I] want someone to actually explain to me fundamentally, what Python is and how it works…in terms of what an environment is, and what it's doing on the back end…what is Anaconda?"

Education

- 15/20: lack of education is a primary problem
  - 10 of these had taken programming classes, but they weren't enough
- Current approaches: workshops (7/20), e.g. Software Carpentry
- But workshops are too shallow: "P10 explained, "[I] want someone to actually explain to me fundamentally, what Python is and how it works…in terms of what an environment is, and what it's doing on the back end…what is Anaconda?"
- Low self-confidence may lead to desire for more formal education, which is unavailable (CS courses aren't a good match)

Technical Skills

Technical Skills

- Besides domain-specific skills (e.g. libraries)…

Technical Skills

- Besides domain-specific skills (e.g. libraries)…
- Data management (access and format problems)

Technical Skills

- Besides domain-specific skills (e.g. libraries)…
- Data management (access and format problems)
- Unpredictable memory, storage requirements

Technical Skills

- Besides domain-specific skills (e.g. libraries)…
- Data management (access and format problems)
- Unpredictable memory, storage requirements
- Heavy use of visualization libraries: "our histogram libraries are lovingly created…a little bit too much religion is involved in creating histogram libraries."

Gradual Learning Curve

Gradual Learning Curve

- Want: small effort to result in small rewards

Gradual Learning Curve

- Want: small effort to result in small rewards
- In reality: small effort is not rewarded; large efforts result in large rewards

Gradual Learning Curve

- Want: small effort to result in small rewards
- In reality: small effort is not rewarded; large efforts result in large rewards
- GUI to CLI

Gradual Learning Curve

- Want: small effort to result in small rewards
- In reality: small effort is not rewarded; large efforts result in large rewards
- GUI to CLI
- Notebooks to traditional programming environments

Gradual Learning Curve

- Want: small effort to result in small rewards
- In reality: small effort is not rewarded; large efforts result in large rewards
- GUI to CLI
- Notebooks to traditional programming environments
- Local to HPC

Gradual Learning Curve

- Want: small effort to result in small rewards
- In reality: small effort is not rewarded; large efforts result in large rewards
- GUI to CLI
- Notebooks to traditional programming environments
- Local to HPC
- In contrast: Resnick's low floors, high ceilings, wide walls

A Vision

A Vision

- Develop languages and tools that afford a gradual progression from small-scale analysis to complex software development

A Vision

- Develop languages and tools that afford a gradual progression from small-scale analysis to complex software development

- Develop and validate scientific software engineering practices and tools

A Vision

- Develop languages and tools that afford a gradual progression from small-scale analysis to complex software development
- Develop and validate scientific software engineering practices and tools
- Facilitate gradual learning via low-commitment formal education

A Vision

- Develop languages and tools that afford a gradual progression from small-scale analysis to complex software development
- Develop and validate scientific software engineering practices and tools
- Facilitate gradual learning via low-commitment formal education
  - Compare MOOCs (many small videos) to semester-long courses (monolithic)

PL Ideas To Consider

- Is functional programming a good match?
  - Can we make a functional PL that "looks" like Python?
  - Closer match to papers (math)?
- Auto-parallelization (for relevant programs)
- Automatic resource estimation (memory, storage, CPU)
- High performance (no GC or rare GC)

Conclusion

- Appropriating SE practices & tools has led to guilt and challenges engaging with the SE community
- But these practices may be inappropriate for many scientists
- Tool and practice designs centered around gradual adoption may make scientists much more effective

Conclusion

- Appropriating SE practices & tools has led to guilt and challenges engaging with the SE community
- But these practices may be inappropriate for many scientists
- Tool and practice designs centered around gradual adoption may make scientists much more effective

Michael Coblenz
mcoblenz@ucsd.edu